NUMBER 41 NOVEMBER 1995

MUJ MACROMEDIA USER JOURNAL

http://www.hyperstand.com

Imaging Databasics with Director

Organization and Tight Lingo Yields Superb Image Databases

By Chuck Henderson

Poor image quality, glitched animations and sloppy transitions can confuse an audience and distract from the drama of a multimedia presentation. In this article, we will examine several strategies for handling large quantities of full-screen, high-quality photographs (images) as one component of a multimedia production. Of key concern is the conservation of valuable disk real estate, the quality of the imagery, the speed of image loading, the quality of transitions, and the ease with which new images can be added to a Director movie. We will discuss and demonstrate techniques for color-correcting and compressing imagery, building photographic data structures and developing efficient and modifiable Lingo code to fetch imagery.

Image Formats and Disk Space

Until ISDN (at 128KBps) or higher-capacity lines become the de facto standard for end-user Internet connections, CD-ROM will be the preferred way to distribute high-quality, full-screen images to personal computer users. One CD-ROM can store approximately 650MB, and these can stream to RAM (and thereby on-screen video memory) at transfer rates between 300KBps for double-speed and 600KBps for quad-speed CD-ROM drives. As we will show, even the ubiquitous double-speed CD-ROM drive can load a full-scale image in less than two seconds under most circumstances. As a component in a multimedia presentation, photographs (and static images of all kinds) can therefore play a significant role.

Images usually start out in 24-bit RGB color, in which 8 bits (or 256 intensity values) are reserved for each of the component colors: red, green and blue. From 24-bit RGB color, the image can be resampled to 8-bit color, in which each pixel becomes a pointer to one of 256 discrete colors in a custom look-up palette. This 8-bit "palettized" color is the most common type in multimedia productions. By reducing a 24-bit RGB image to an 8-bit image, we reduce our disc-space storage requirement by at least one-third and our image loading time by one-third as well! Most important, if the custom look-up palette's colors are carefully chosen to match the colors in the original RGB image, and if a dithering scheme is used whereby adjacent pixels toggle back and forth between two palette colors to create an "in-between" or "dithered" color, then the user, in most instances, cannot distinguish between the 24-bit RGB image and the properly dithered, 8-bit palettized image. The careful construction of high-quality 8-bit images that have a well-chosen color palette is key to quality images in a CD-ROM product. The details of building high-quality 8-bit images from 24-bit RGB images will be discussed shortly.

If we consider a full-screen image to occupy 640 by 480 pixels (standard 13-inch monitor resolution), then each full-screen, 8-bit palettized image will require about 307KB. But since most images have considerable inter-

(continued on page 4)

IN THIS ISSUE...

2 **MUJ NEWS:** Eric Elia reports back from the Macromedia User Conference. Shockwave, Java, Director 5 and more...

8 **ONLINGO:** Part 2 of Kurt Cagle's look at the Movie Utilities XObject.

13 **AUTHORWARE:** Tom King wades through a sea of Authorware resources.

15 **RESOURCES:** *Director Demystified,* the best Director book around.

16 **VERSION WATCH**

Publisher: HyperMedia Communications Inc. Not affiliated with Macromedia Inc. Macromedia is a trademark of Macromedia Inc.

Back from the Show

By Eric Elia

In the wake of 1995's major new product releases, alliances and acquisitions, the Macromedia User Conference was bigger and badder than ever before. Bud and Co. waited for the big annual event, held this year at San Francisco's Moscone Center, to deliver their biggest thunder yet. A major push to bring multimedia to the Internet, new alliances, brand-new technology, a major upgrade for Director in '96 and a cross-application standardization initiative kept Macromedians walking around with their mouths agape for three days.

Alliances

After Bud Colligan, president and CEO of Macromedia, delivered his keynote, Macromedia stock went up 6 points. Some attributed this fiscal enthusiasm to the announcement of a new cooperative relationship between Sun Microsystems and Macromedia regarding Java, Sun's new network-ready programming language that brings dynamism to static Web pages. Java has generated a great deal of press lately because it will allow information to be updated constantly across the World Wide Web through small, multithreaded Java applications, or applets, whether that information exists as text data, graphics or other media.

The two companies will build support for Java applets into Director and Authorware, and develop multimedia-class libraries to build into future versions of Java. In addition, Macromedia will integrate Java into a next-generation authoring tool for high-bandwidth, Internet-based multimedia, leveraging today's core multimedia authoring technology. "By combining Sun's open-standards programming and server expertise with Macromedia's leadership position in new media, we can offer new functionality and tools to our customers," said Colligan.

On any other day, the Macromedia/CompuServe announcement would have been bigger news. Macromedia and the number one or two (depending on who you ask) online service will expand Macromedia's fabled "author once, play anywhere" initiative, bringing Director Player technology to the CompuServe client software. The technology will first appear in a new version of CompuServe's online service, code-named WOW!

Shockwave

If you asked the developers at the show, the real killer announcement was Shockwave, new technology from Macromedia and Netscape that allows Director movies to be played through WWW browsers that include Netscape Navigator 2.0, followed by browsers from Navisoft, Microsoft and Silicon Graphics. This opens up a whole new opportunity for delivering interactive media on the Web and gives creators new interface design potential. "The Internet is quickly evolving from a static, text-based medium to an interactive broadcasting network, complete with multimedia content," said Lee Swearingen, a partner at DXM Productions and an early Shockwave user. "With Shockwave technology, we will be able to instantly publish our multimedia productions for

Be sure to visit Hyperstand, our home on the Web, for a look at Shockwave in action (http://www.hyperstand.com).

MUJ MACROMEDIA USER JOURNAL

http://www.hyperstand.com

EDITORIAL DIRECTOR: Phil Hood
EXECUTIVE EDITORS: Tony Bové and Cheryl Rhodes
MANAGING EDITOR: Eric Elia
CONTRIBUTING EDITORS: David Biedny, Kurt Cagle, Gary Gattis, Chuck Henderson, Tom King, Lynda Weinman
COPY EDITOR: Pat Soberanis
ART DIRECTOR: Nancy Cutler
ASSISTANT ART DIRECTOR: Jake Flaherty
ART INTERN: Agnes Chan
PUBLISHER: Penny Ertelt
CIRCULATION ASSISTANT: Frank Ha

Subscriptions are $175 ($200 outside the U.S.) for 12 issues, $250 ($275 outside the U.S.) for 24 issues. Back issues are available upon request. Single copies are $12, or $6.50 each for 10 copies or more.
Direct all subscriptions, sample copy and back-issue requests to: *Macromedia User Journal*, P.O. Box 469079, Escondido, CA 92046-9779. Voice: (800) 546-6707 (9:00 a.m.–5 p.m. PST) Outside the U.S., call (415) 573-5170, ext. 249. Fax: (619) 743-2590 MUJSub@newmedia.com.
Send article submissions and letters to: *Macromedia User Journal*, 901 Mariner's Island Blvd. #365, San Mateo, CA 94404. MCI Mail and AppleLink: newmediamag; CompuServe: 70640, 2300; America Online: MUJ Editor; Internet: MUJEditor@newmedia.com.
Tony Bové: 30777 S. Hwy, 1/P.O. Box 1289, Gualala, CA 95445. MCI Mail: TBOVE; CompuServe: 70105,722; AppleLink: D0493; America Online: TonyBove; Internet: tony@rockument.com.
Macromedia Director, Macromedia, MediaMaker, Macromedia Three-D, Lingo, Windows Player and XObjects are trademarks of Macromedia Inc.
This publication is not affiliated with Macromedia Inc., nor is Macromedia responsible for its contents.

Macromedia User Journal is published 12 times a year by HyperMedia Communications Inc. Copyright ©1995 by HyperMedia Communications Inc.
(ISSN 1065-3929)

Got a question about your Macromedia User Journal subscription? Call (800) 546-6707 for questions regarding billing and delivery.

a global audience. We can easily update and exchange Director movies to keep sites dynamic. This will keep people coming back to those sites."

The Shockwave development kit will be available from Macromedia for Windows 3.1/95 at http://www.macromedia.com after the final release of Netscape Navigator 2.0 this month, and will consist of Afterburner, HTML documentation, sample movies, browser plug-ins to play the movies and "Made with Macromedia" guidelines (optional for Web pages).

Afterburner, a post-authoring compression utility, and other included compression utilities can bring file sizes on a Director movie down approximately 60 percent. A demo of Apple Computer's home page was shown with a revolving 3-D apple logo and hot items with alternating rollover graphics. The file size of the movie was actually smaller than the original graphic. Tricks for bringing file sizes down include using Lingo to colorize 1-bit cast members or to shift palettes for animation, rather than using new or alternate graphics images with low bit depth, linking to multiple movies, and using small looping audio and animation.

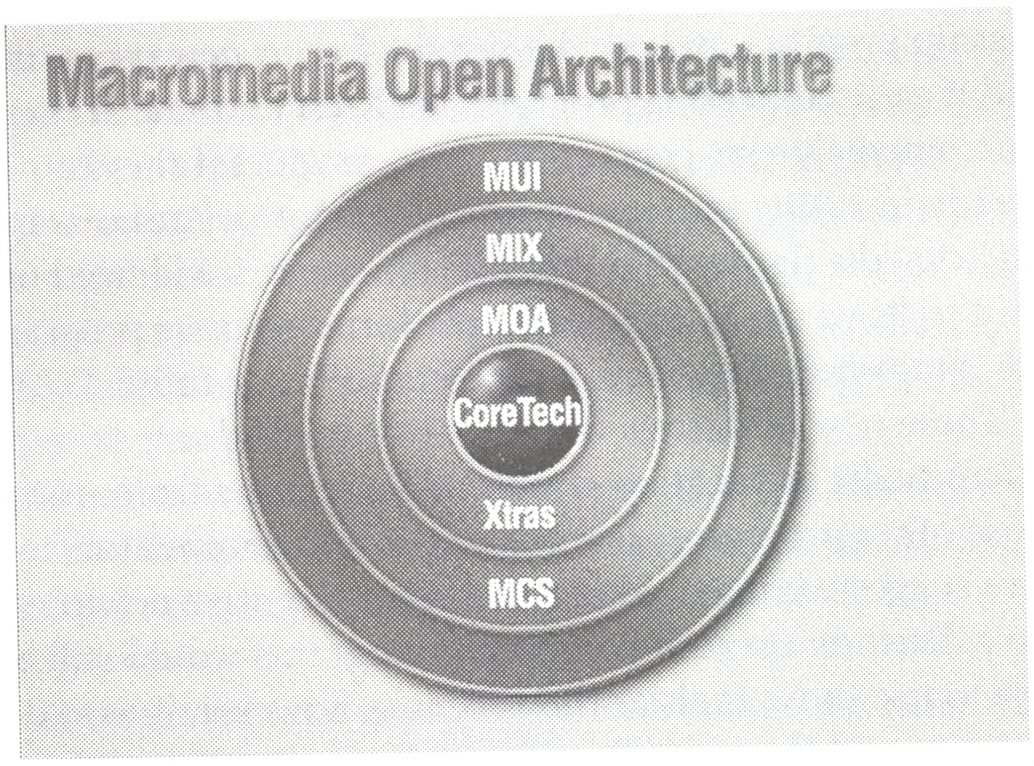
A few new Lingo commands will be available that allow Directorites to link parts of a movie to different URLs. New HTML will be used to embed movies within an HTML document. Shockwave won't support fileIO or Lingo commands such as shutDown, for obvious reasons. Future versions will include support for Movies in a Window, the ability to move directly to a specific frame of a movie, and streaming movies so they don't have to be downloaded fully before they start playing.

Rumor has it that Shockwave for Authorware and FreeHand are on the horizon.

The Macromedia Platform

Macromedia presented its vision of the "Made with Macromedia" future, a cross-platform, cross-application standardization initiative that would make it easier for new users to work beyond the core Macromedia program they purchased. Imagine purchasing Authorware, and getting with it Director, Extreme 3D, SoundEdit 16, Xres 2 (remember the Fauve acquisition, *MUJ*, October 1995) and FreeHand, and finding similar interfaces with the same menu bars, core palettes and tool bars, a common scripting language based on Lingo for all the programs and plug-ins or "Xtras" that work in more than one program. The new menu bar will have File, Edit, View, Insert, Modify, Control, Xtras, Window and Help commands.

The platform is based on five elements. The first is the Macromedia Core Technology, which is basically the guts of each program. Second is the Macromedia Open Architecture (MOA), an API for third-party developers to create plug-in Xtras that extend the power of Macromedia

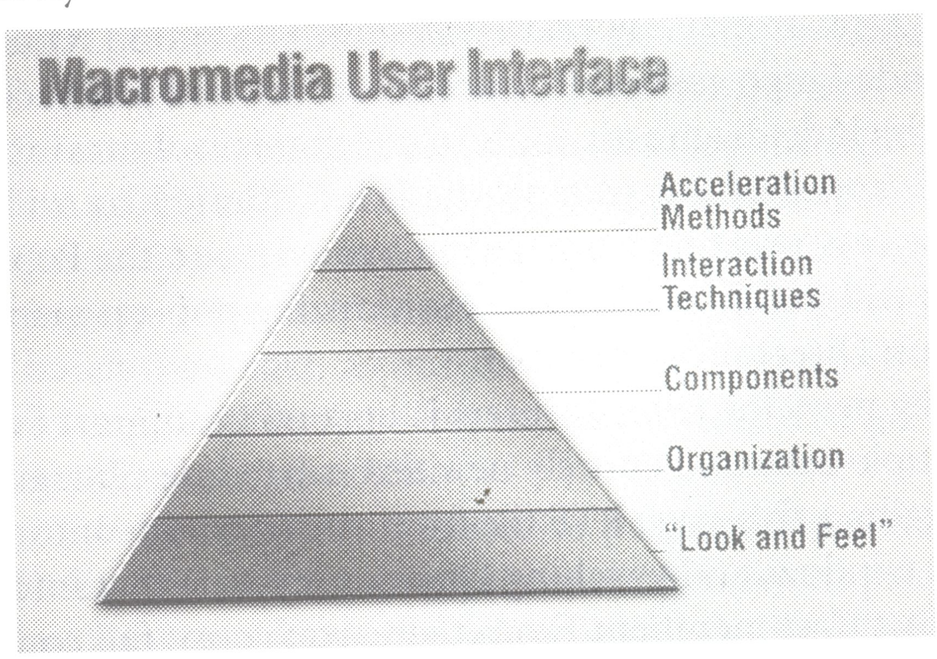


Core Technology. Xtra Developer Kits (XDKs) are now available to Director 5 and SoundEdit 2 beta sites. MOA will also allow developers to create Xtras that work in more than one application, such as transitions for both Authorware and Director.

Third is the Macromedia Information Exchange (MIX), a file-format steamroller that allows document sharing between Macromedia applications. Imagine creating an outline image in FreeHand, dragging that outline to Extreme 3D, extruding it and mapping a texture created in Xres, and then dragging the rendered model to Director 5, without all that clunky importing and exporting.

Fourth is the Macromedia Common Scripting (MCS) language based on Director's Lingo. MCS will enable automation, such as batch processing of audio files in the next SoundEdit, and runtime programmability. This will give developers more control over programs such as FreeHand, SoundEdit and Extreme 3D, and leverage the Lingo knowledge they've already gotten from Director.

Finally, there is the Macromedia User Interface (MUI), a standardized interface across the whole product line that will use the standard menus, tool bars and palettes. MUI fits nicely into the Macromedia Studio metaphor—buy a



In the Macromedia User Interface pyramid, application interfaces will be most similar in their look and feel, and least similar in application-specific acceleration methods such as shortcuts.

flagship program and get the rest for free. A new user will find it easier to learn a program like Xres, Extreme 3D or Macromedia's next-generation video product if they're already investing the time to learn Director or Authorware. Macromedia figures that this way, fewer users will need or want to leave the Macromedia "platform" to use a program like Photoshop, Premiere, Strata StudioPro or Infini-D.

These new initiatives would not close off the Macromedia application circle and limit users from using those other programs, as some critics and competitors have suggested; it would just create an incentive not to leave. Existing users of competing programs would still be able to use those tools as they do now, however, it would be easier and the work-flow more efficient to stay within the Macromedia platform. Macromedia seems to be staking its future on that notion.

Director 5

Drum roll, please: Director 5 gives us much to be excited about in the coming year. Due in early 1996, Director 5, affectionately known as Spike, will pack a host of new features that users have been clamoring for, including many of the standardization features discussed above. Of particular interest is MOA, the new open architecture for plug-in Xtras.

At the conference, an incredible demo of a QuickDraw 3D Xtra for Power Macs was shown; it treated a true 3-D object as a regular old Cast member. When placed on the Stage, it could be rotated in any direction in real time. Textures were created and edited in Director's familiar Paint window, and then remapped perfectly. Support for live video—without extra programming—was demonstrated as well. When the live video was mapped to the 3-D object in Director, in real time, you could hear gasps in the audience.

Other Xtras include MIDI device control, Movie in a Window power from a simple pull-down menu, and three new Xtras from Gray Matter Design: The Sound Xtra will provide Lingo control to voice recording and playback, and include diagnostic tools to check microphone input levels and sound-card type; the PrintOMatic Xtra will allow easy printing from within a movie; and the PopUp Xtra will make creating and customizing pop-up menus within Director much easier.

Big news for Director users is upcoming support for multiple casts and entirely new text capabilities. Imagine dragging a five-page RTF file from your desktop onto your Cast window, and watching Director separate that into five distinct Cast members. From there, you've got full control over font, size, color, true styles, paragraphs and margins from a junior word processor within Director. To top it off, Spike will have real-time anti-aliasing, eliminating the need to create text in an image editing program and kicking it back into Director. There will also be scrolling text windows and Lingo control over those windows.

Performance enhancements for Director 5 include a major preloading upgrade that supports multiple movie preloads, making branching to associated movies much faster. With this new feature, a few-second delay was eliminated entirely on just a Pentium 66 with a quad-speed CD-ROM drive.

Expect plenty more news in all of these areas, particularly regarding Director 5 and the new SoundEdit. We'll be sure take a closer look at the new open architecture and common scripting language, Shockwave, Xres 2 and Macromedia's video product, and we'll update you on Director 5 as more new features are let out of the bag.

Macromedia Inc. (415) 252-2000
http://www.macromedia.com

Gray Matter Design (415) 243-0394

(PHOTOGRAPHIC DATABASES: from page 1)

image "spectral redundancy" (adjacent pixels have the same value), they can be compressed to a significant degree. The PICT format supported by Director automatically provides what is called run-length encoding, which in most instances compresses full-screen PICT images to less than 200KB. Generally, the simpler the image, the smaller the disc space required to store the compressed PICT image. (Run-length encoding represents data as a pixel color value followed by the number of pixels in the scan line that have the same color value.)

Editing and Color-Correcting Images

If you are scanning your own photography (either slide or print), you will gain the greatest productivity by scanning directly into an image editing application such as Adobe Photoshop. Most scanning hardware include driver software that are designed as Photoshop plug-ins; if so, you can access the scanner driver in the File/Acquire submenu. Scanning directly into Photoshop enables you to immediately crop, resize and color-correct the RGB image.

Color correction is a much-discussed issue among graphic artists throughout the design community. There is a world of difference, however, between color-correcting imagery for multimedia presentations and for color printing. In the former, the RGB rendition on the screen corresponds exactly to the finished product (i.e., the product will ultimately be viewed on-screen). In the latter, the image must be converted to CMYK format (for the four printer's inks: cyan, magenta, yellow and black) or

matched with custom inks such as PMS (Pantone Matching System) colors. For multimedia products shipping on CD-ROM, we are interested in only screen-ready imagery.

Unfortunately, the great majority of users' monitors are completely maladjusted for color balance, contrast and brightness. Furthermore, users often position their monitors so that the glare from a light bulb or, worse, the errant rays of the sun itself bounce off the screen, destroying the color fidelity of the image designers have worked so hard to perfect. Until media literacy is a required course for all high-school freshmen, there is no real solution to this malady. I would, however, suggest that you license and ship the Gamma monitor-calibration utility with your product (it comes with Photoshop). By instructing your users how to position their monitors properly and, using Gamma, how to adjust them, you will help some users appreciate more fully the quality imagery you have toiled to prepare.

Selecting a Palette Strategy

One of Director version 4.0.4's fault lines occurs in the maximum number of custom palettes that can be included in a single movie. That number is 54. The 54-color palette limitation did not exist in previous versions and, according to my sources, will be removed in future versions. Because of the radical changes Director 4 inflicted on version 3, limitations in the number of palettes have been "temporarily" inserted. Nevertheless, 54 palettes is enough to accommodate several thousand slides with excellent results.

If your image library follows a common theme (say, beaches, wildlife, outdoor shots or flower arranging), then the range of colors for the slides in each domain will probably be clustered in similar parts of the full range of the color space. By organizing your slides in logical color groups—with all the slides in any one group sharing similar color, intensity and brightness characteristics—you maximize the likelihood that a "superpalette" (a palette that is tailored to many images) can be satisfactorily determined.

DeBabelizer to the Rescue

Without Equilibrium's DeBabelizer, Macromedia's stock value would probably be 10 points lower than it is today—just from the loss in productivity and image quality. As it is, DeBabelizer's many batch-oriented capabilities allow us to make sense out of a bewildering assortment of image formats—and, more to the point, to build superpalettes.

For example, if you were planning on including 1,500 full-screen photographs in your CD-ROM project, you could easily divide these among, say, 50 palettes (that gives you four extra palettes for other special requirements of your CD-ROM, which would bring you to the 54-palette limit). So each superpalette would service approximately 30 different images on average. More likely, the number of images assigned to each palette would vary, so that some palettes would service 50 or more images, while others would service just five (or fewer) particularly sensitive images.

My own work with the following technique has delivered excellent results, even when images using the same superpalette are quite diverse in original color composition:

1) Organize each grouping of RGB images by palette by place them in separate folders.

2) Using DeBabelizer, create a superpalette for all of the image files in each of the folders.

3) Create an empty folder for each of the RGB image folders.

4) Run DeBabelizer's Dither to Superpalette command in batch mode, instructing the program to place the new 8-bit dithered image for each of the RGB images in the new folder.

The dithering in an 8-bit image only becomes visible in large image areas that exhibit very subtle changes, such as a sky that shifts from one shade of blue to another. In that case, the dithering effect is visible on close inspection, but the overall sky effect is still quite acceptable. Another anomaly becomes visible when a small area of only one image in a superpalette group has a very unusual or strident color, such as a small red-orange flower. The superpalette may not reserve special colors to accommodate the flower color and, when dithering to the superpalette, may instead make some very bad choices (e.g., the flower ends up looking brown).

The solution to this color-singularity problem is to confine the superpalette construction to less than the full complement of 256 colors and then to insert an additional group of colors in the superpalette to handle special colors such as the red-orange in the flower. Create a range of special colors representing the full spectrum of the color space (reds, oranges, blues, magentas, etc.) and include these special colors as part of the superpalette. Any anomalous colors that show up in the batch files, such as our red-orange flower, will then be composed in the 8-bit domain by dithering between these special colors.

Figure 1 shows an 8-bit, 256-color palette divided into logical areas that will accommodate many different palette considerations, including the color-singularity problem discussed above. In this figure we used only 180 of the 256 possible palette colors to build the initial superpalette, which leaves us with 76 colors for other purposes.

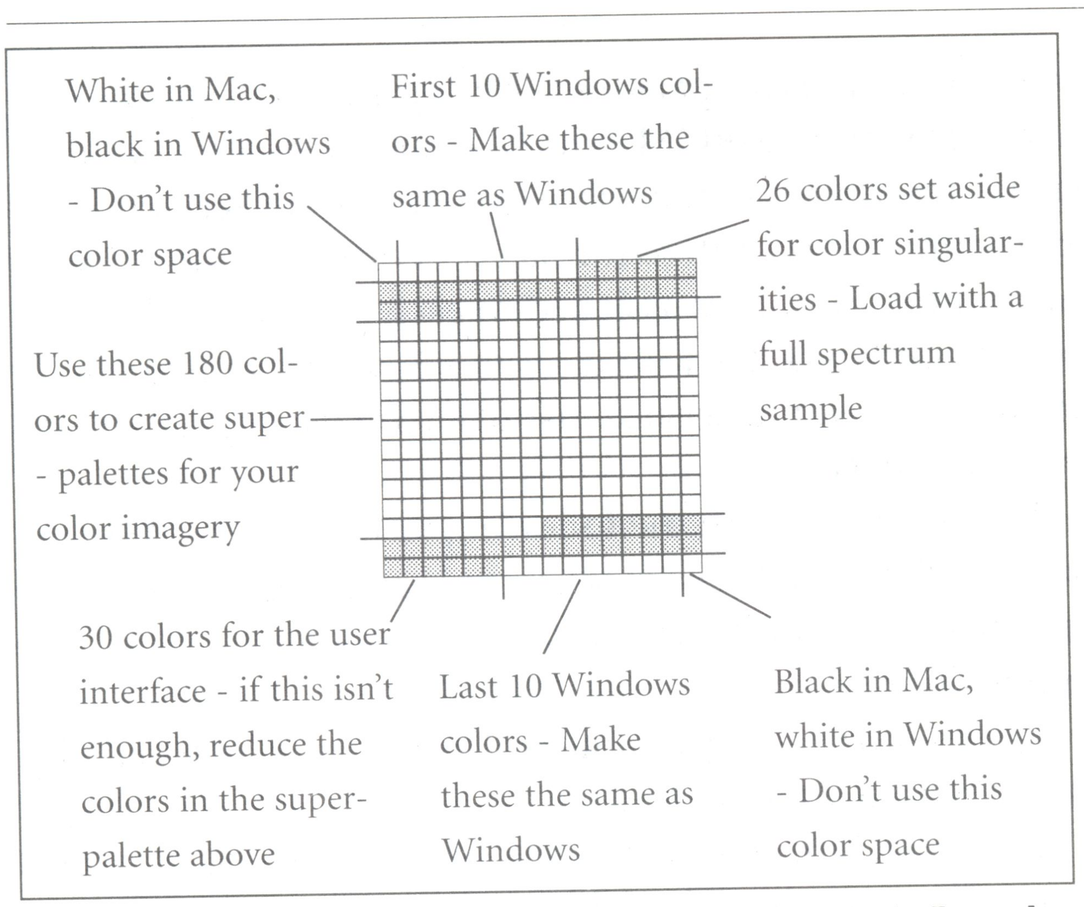

White in Mac, black in Windows - Don't use this color space
First 10 Windows colors - Make these the same as Windows
26 colors set aside for color singularities - Load with a full spectrum sample
Use these 180 colors to create super - palettes for your color imagery
30 colors for the user interface - if this isn't enough, reduce the colors in the super-palette above
Last 10 Windows colors - Make these the same as Windows
Black in Mac, white in Windows - Don't use this color space

Figure 1: Superpalette Divisions. By manipulating the above five subsections of the 256 color palette, you can stay out of trouble with Windows and provide plenty of color space for the unchanging user interface and the custom superpalette. It's OK to use colors 1 through 254 when palettizing an image, avoid colors 0 and 255.

Here's what you can do with these extra 76 colors:

1) If you are building a cross-platform project (and who isn't these days), avoid using colors 0 and 255 in any palette. These positions are always reserved for the colors black and white, and unfortunately their values on the Mac are reversed in Windows.

2) If you are building a robust Windows product, assign specific colors to the first 10 and last 10 color positions in each palette. These are colors that are usually reserved in the Windows environment for standard Windows colors (borders, backgrounds, highlights, etc.) and will be used by Windows if the user breaks out of your application (projector), while leaving it running in the background (with the superpalette still in place). The RGB values I assigned to these color slots in the palette are listed in Figure 2.

Palette Position	Red	Green	Blue	Palette Position	Red	Green	Blue
0	255	255	255	246	255	251	240
1	191	0	0	247	160	160	164
2	0	191	0	248	128	128	128
3	191	191	0	249	255	0	0
4	0	0	191	250	0	255	0
5	191	0	191	251	255	255	0
6	0	191	191	252	0	0	255
7	192	192	192	253	255	0	255
8	192	220	192	254	0	255	255
9	166	202	240	255	0	0	0

Figure 2: RGB values for the first and last 10 color slots in the superpalette.

3) With the 56 colors that are still unassigned, take approximately 26 and assign colors for the singularities that will show up in small portions of some images (such as the red-orange flower described earlier).

4) You can assign the remaining 30 colors to the user-interface components and will remain constant across all of the palettes in your movie.

5) Palettize each image using all of the colors in the palette, except for colors 0 and 255.

Thankfully, DeBabelizer gives us all the tools we need to create superpalettes and to palettize using the criteria suggested above. For more information about DeBabelizer strategies, see "DeBabelizer and Director: A Strategy for Palette Management" by Patrick Quinn in *MUJ*, January 1995.

Movie Organization and File-Naming Conventions

Following the above procedure, we now have in place a series of folders, each containing an arbitrary number of 8-bit image files, and all of the files in each folder have been palettized to a unique superpalette. I will now explain a file-naming convention that allows the parsing capabilities within Director to evaluate an image file by name, install the correct palette for each image, and load the image to the screen.

Figure 3 shows the rudimentary user interface, used in a movie to demonstrate this image management technique for accessing a large photographic database.

Figure 3: The simple user interface used in a movie demonstrating this image management technique.

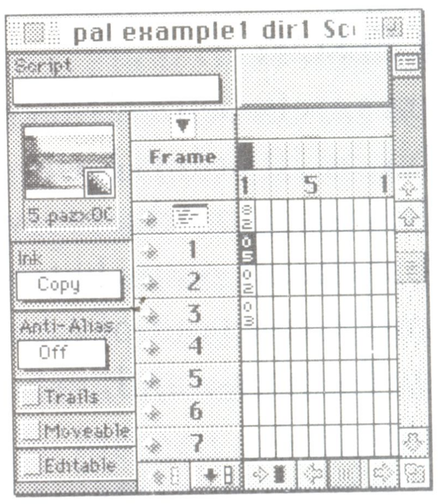


Figure 4: A small score for accessing any image in the database.

In this Director movie the following characteristics should be noted:

- Image files are named so that the last three digits in an 11-character file name indicate the appropriate palette (see Figure 5).

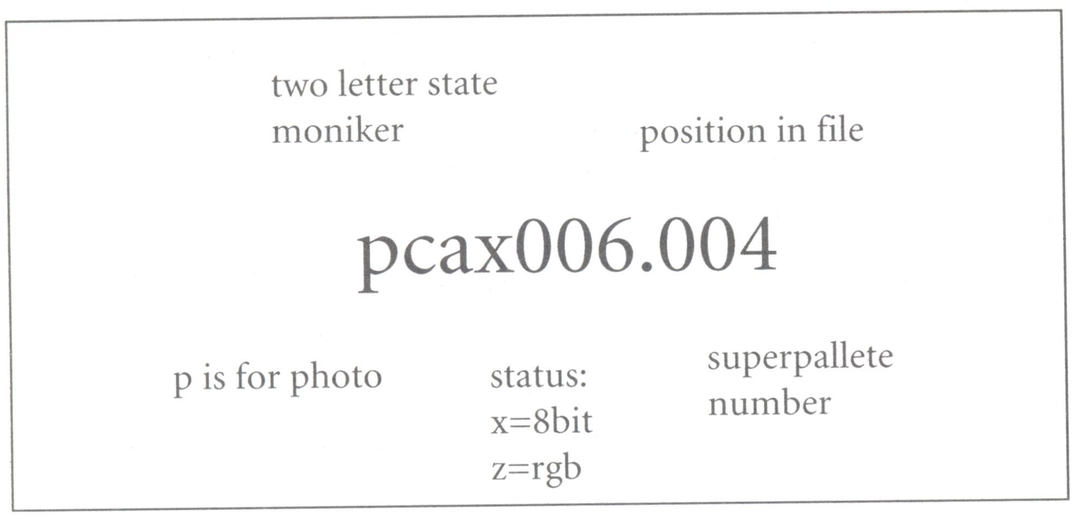


Figure 5: An Example Image-Naming Scheme. By embedding information in the file name itself, the name can be parsed and interpreted by straightforward Lingo code. Just like a real "object," Director can determine that the file is an image, solve for the right palette and load the palette prior to displaying the image. (Always keep your file names legal in the Windows environment; eight characters maximum with a three-character extension.)

- The cast (a portion of which is shown in Figure 6) is made up primarily of the 8-bit PICT images—32,000 cast members are possible in Director (all of the images are imported by selecting File/Import, with Link to File turned on).

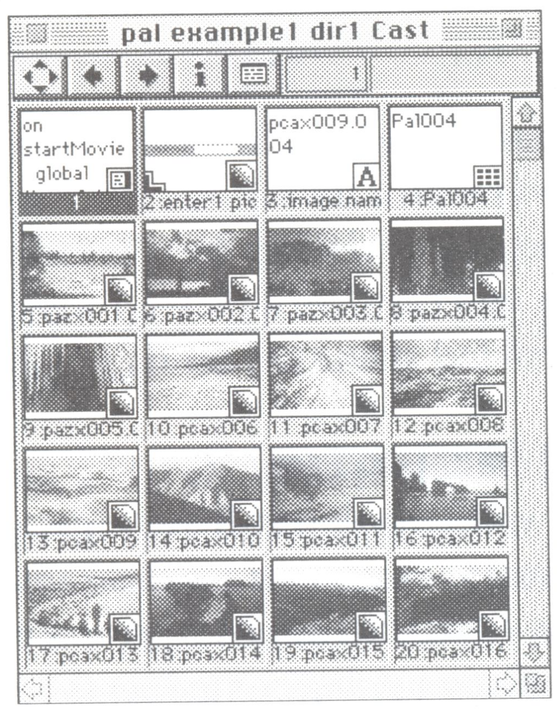


Figure 6: A portion of the cast.

- The interface (see Figure 4 above) allows the user to type in the name of the image file and then press the Get It button. In less than one second—for most quad-speed CD-ROM drives—the new image is posted on the screen.

This interface is rudimentary, but what we're trying to demonstrate here is really a back end to any user action that might determine the appropriate "next image," such as clicking on a map, jousting with a character in a dramatic presentation or typing a category, such as "wildflowers," and letting a database routine parse and search for the right image.

Lingo for Palette Manipulation and Image Display

The following Sprite Script belongs to the Get It button. When the button is clicked, the simple process of retiring the old image, switching palettes (if needed) and loading the new image is accomplished.

```
on mouseUp
  global the.palette
  global image.num

  set the visible of sprite 1 to FALSE
  updateStage

  put "Pal" into the.palette
  put char 9 to 11 of field "image.name" after
the.palette

  puppetPalette the number of cast the.palette
  set the castNum of sprite 1 to the number of
cast the text of cast "image.name"
  set the visible of sprite 1 to TRUE
  updateStage
end
```

Let's review the above code, which is attached to the Get It button (sprite 2).

```
global the.palette
global image.num
```

Here we have set the two variables *the.palette* and *image.num* to blank (i.e., no text) via the command *set the.palette to* "" in the `on startMovie` handler. So we are obliged to declare them as global variables wherever they are used in all of the scripts in the movie.

Usually, good Lingo style would dictate that global variables begin with a *g*. So *g.the.palette* and *g.image.num* would, perhaps, have been better name selections. However, like most rules, the "g-for-global" rule can be broken and often is, even by the most prodigious Lingo gunslingers.

```
set the visible of sprite 1 to FALSE
updateStage
```

By turning off the visibility of the image sprite (sprite 1), we spare the user any notice that the palette is about to change. With the background set to black, the image simply disappears while we do the palette change. Don't forget to update the stage, the updateStage command is what implements the new visibility assignment. If you know that the palette won't be changing, there's no need to

make the sprite invisible; the change from one image to another will happen instantly on the screen, with no black between the image paints.

```
put "Pal" into the.palette
put char 9 to 11 of field "image.name" after
the.palette
```

The above code extracts the correct palette name from the image file name. The naming convention for the image files is such that the palette number is the suffix in the file name (e.g., pcax005.004, where 004 is the palette identifier). Because all of the palette names in the cast have been set to begin with "Pal" (as in Pal004), we assign the correct palette by first inserting "Pal" into the variable *the.palette*. Since the user has specified the image file name by typing into the editable text cast *image.name*, we can interrogate this variable and, by extracting characters 9 to 11 from its contents, isolate the palette number. Then we append these three numbers after "Pal" in the global variable *the.palette*, thus successfully isolating the palette name.

```
puppetPalette the number of cast the.palette
set the castNum of sprite 1 to the number of
cast the text of cast "image.name"
```

We now switch the palette. The puppetPalette command can be invoked at any time to both make the current palette a puppet and to simultaneously change the assigned palette. Instead of referring directly to the cast number of the palette, we point to *the number of cast the.palette*. To return the palette to the control of the Score, you would invoke `"puppetPalette 0"`. Since the visibility of sprite 1 is turned off and the colors of the remaining objects on the stage (the user-interface elements) are assigned to colors that don't change from one palette to another, the user is completely unaware that a palette change is happening.

The next line does the work of assigning the correct image to sprite 1. Since all of the images have been preloaded into the cast, we use the very fast set the castNum assignment command to swap in new images. Because we don't change frames and don't need to turn the visibility of sprites on and off (as in the case where each image would occupy a cell in the Score), our code is extremely compact. An entire CD-ROM filled with 3,000-plus full-screen and superpalettized images can, in this example, be accessed by manipulating just one cell in the Score. (For more information about using castNum assignments, see "Glitch-Free Animation" in *MUJ*, September 1995).

```
set the visible of sprite 1 to TRUE
updateStage
```

The last step in this process is to turn on the visibility of sprite 1 again. The only difference is that the palette has been changed and a completely new image has been loaded from the CD-ROM. As before, update the stage (via updateStage) to see the new image.

Last Word

Any photograph or image from a large library of over 3,000 images stored on a single CD-ROM can be made visible to users in about one second. By carefully grouping similar images by color and tonal differences, and by building custom superpalettes for each color group, the presentation of these images can rival true 24-bit RGB imagery. By using the castNum assignment technique, a few lines of Lingo code can provide immediate access to any imagery in the cast.

Chuck Henderson (earthmap@mcn.org) is a multimedia author living in Gualala, California.

onLingo

The Lingo Utility Belt: The Movie Utilities XObject, Part 2

By Kurt Cagle

Last month, we continued our discussion of XObjects with a look at the Movie Utilities XObject, particularly system information, graphics, and the stage and cast. We'll pick up where we left off with a look at text and string manipulation in the XObject.

Text and String Manipulation

If you've played with Lingo's text-handling capabilities, you're probably aware of the curious dichotomy that exists there. On the one hand, there is a surprisingly robust mechanism for manipulating string components—the ability to change the itemdelimiter is a prime example, as is the ability to format text within a text box, an element that requires considerably more work in such languages as Visual Basic. On the other hand, there are times when doing the simplest tasks, such as changing a lowercase character to an uppercase one, requires you to go through all kinds of nasty hoops. Several of the routines in the text-handling section are things that can be done in Lingo, but that's considerably slower and takes many more lines of code.

For example, consider the problem of converting a text

string so that the first character in every word is capitalized and every subsequent character is lowercase. The Movie Utilities XObject provides two very handy routines, the mToUpperCase and mToLowerCase, that are tailor-made for such situations. We can use them to create a ToMixedCase handler that looks something like this:

```
on ToMixedCase myStr
  openxlib "movutils" -- opens the movie utilities XObject if it's not already open
  set utils=movutils(mNew) -- create an instance of the XObject
  set target="" -- create a target string
  set tempchar="" -- and a temporary store for the characters
  repeat with index=1 to the number of words in myStr -- for each word in the string
    set myWord=word index of myStr -- get that word
    repeat with charcount=1 to length(myWord) -- iterate over the word's length
      set tempchar=char charcount of myword -- get the succeeding character
      if charcount=1 then -- if it's the first character then
        set tempchar=utils(mToUpper,tempchar) -- make it upper case, if possible
      else -- otherwise
        set tempchar=utils(mToLower,tempchar) -- make it lower case, if possible
      end if
      set target=target&tempchar -- add the character to the target string
    end repeat
    set target=target&" " -- add a space to separate the words
  end repeat
  delete char (the number of chars in target) of target -- delete the final space
  utils(mDispose)
  return target -- return the target string
end ToMixedCase
```

This method saved about four steps. Without it, a mechanism would have had to be in place to specifically check to see whether each character fell within a given range, convert the character to an ASCII value, and then add or subtract an offset to that value to get the new characters. This ToMixedCase routine also runs a little faster, since the searching and testing routines are coded into a compiled language that's been optimized for performance.

The other conversion routines are of varying utility; mTrimChars can be used to remove the leading and trailing spaces from a string; this can be useful when attempting to pad a word to a specific number of characters. The mNoPunct handler removes all punctuation from a string, although it doesn't remove spaces and actually replaces punctuation characters with blank spaces rather than properly deleting the punctuation from the string.

The handler mDollarFormat is fairly useful because it can do something that is remarkably difficult to do in Lingo: It will take a floating-point or integer number and convert it into U.S. currency format. For example, consider the following:

```
set myval=3224.54+223752.09
put utils(mDollarFormat,string(myval))
==> "$226,976.63"
```

If the value passed to the function is a floating-point number, it will have the accuracy of the floating-point number with a minimum of two digits after the decimal point. The largest value that can be reliably used with this function is $2 billion ($2,147,483,647 to be precise), which by some strange coincidence happens to be the maximum integer currently allowed on the system. Note that the function expects to be passed a string, not a number. Numbers less than zero are represented in bookkeeping notation; that is, the numbers are shown as positive values within parentheses.

```
put utils(mDollarFormat,string(-23834))
==> "($23,843.00)"
```

Lingo deals with text in chunks—characters, words, lines, items and so on. One of the problems that occasionally occurs when dealing with text is that there is no effective way to determine the actual character position where a given word or line starts. The offset command will tell where a given substring is within a string, but it's much more difficult to determine which word was actually chosen from an offset value. The Movie Utilities XObject provides four functions that, while perhaps not as intuitive as they could be, offer at least some solution.

The mGetWordStart and mGetWordEnd methods take as arguments the string to be searched and the index of the word (i.e., how many words from the beginning of the string to that word). From this it returns the character index indicating the start and end of that word, respectively. At first, these functions threw me, because the documentation implies that they are search functions, yet their operation is at a slightly lower level. As an example (albeit one with limited real-world utility), consider the following function, which takes a selection of text, and a substring that may or may not be a word, and returns the text and the positions of all words that include that substring in the selection.

```
on FindWord SrcStr,SrchText
  openxlib "MovUtils"
  set utils=movutils(mNew)
  set BaseWordOffset=0
  set textptr=offset(SrcStr,SrchText)
  set foundlist=[:]
  repeat while textptr>0
```

```
        set numwords=the number of words in SrcStr
        repeat with index=1 to numwords
           set
EndPtr=utils(mGetWordEnd,SrcStr,index)
           if EndPtr>=textptr then
              set
StartPtr=utils(mGetWordStart,SrcStr,index)
              set
foundlist=[BaseWordOffset+index,char StartPtr to
EndPtr of SrcStr]
              delete word 1 to index of SrcStr
              set basewordoffset=index
              index=numwords+1
           end if
        end repeat
        set textptr=offset(SrcStr,SrchText)
     end repeat
     utils(mDispose)
     return foundlist
  end FindWord
```

This function takes the string, uses the offset function to determine where the position is in the text of the substring, and then iterates until the starting and ending pointers provided by the Movie Utilities method surrounds the other pointer. Then it places that word into a property list, along with the position of that word in the string. After this, everything up to the word found is removed from the string, and the process is repeated until the substring is no longer found in the search string.

A similar and perhaps more useful function can be created using the two other methods, mGetLineStart and mGetLineEnd. For those familiar with UNIX, these methods can be used to approximate a grep-like capability. The mGetLineStart and mGetLineEnd methods return the starting and ending positions of the indexed line. A distinction should be made between the lines that are displayed in a text box and the notation of "line" in Lingo. A text box will wrap text around to the next line beneath it, but if queried, the wrapped word is still on the same line as the previous text. A line in Lingo is terminated by a carriage-return character, regardless of how the text wraps in a text box. This is unfortunate, since it makes implementing a scrollable text box much more difficult.

A moment's inspection of the following code should reveal that the only difference between the two routines is that *word* has been replaced by *line*. Other than that, FindWord and FindLine are identical.

```
on FindLine SrcStr,SrchText
   openxlib "MovUtils"
   set utils=movutils(mNew)
   set BaseLineOffset=0
   set textptr=offset(SrcStr,SrchText)
   set foundlist=[:]
   repeat while textptr>0
      set numLines=the number of Lines in SrcStr
      repeat with index=1 to numLines
         set
EndPtr=utils(mGetLineEnd,SrcStr,index)
         if EndPtr>=textptr then
            set
StartPtr=utils(mGetLineStart,SrcStr,index)
            set
foundlist=[BaseLineOffset+index,char StartPtr to
EndPtr of SrcStr]
            delete line 1 to index of SrcStr
            set baseLineoffset=index
            index=numLines+1
         end if
      end repeat
      set textptr=offset(SrcStr,SrchText)
   end repeat
   utils(mDispose)
   return foundlist
end FindLine
```

You may have noticed a curious phenomenon when taking text files created on the Macintosh and porting them over to Windows or vice versa. When read on the Macintosh, Windows text files frequently lose their carriage returns, making perfectly formatted text a mishmash of extremely long lines. On the flip side, text files carried from the Mac side to Windows can show up with odd little boxes that you *know* weren't in the original Mac text. The reason for this is simple, albeit maddening: On the Macintosh, the end of a line in a text file is demarcated by the sequence Carriage Return/Line Feed, while the PC only recognizes Carriage Return (this isn't consistent across all word processors, many of which automatically convert from one form to the other invisibly).

Two routines that can help in this translation are the mCRtoCRLF (for adding line feeds to carriage returns when going from Windows to the Macintosh) and mCRLFtoCR (for going the other way). Each of these methods takes a string as an argument and returns a string.

```
  SS   mCRtoCRLF        —Add a LineFeed to each
Return char within Macintosh text
  SS   mCRLFtoCR        —Remove LineFeed from each
end of line within Windows text
```

The final set of string functions are all nearly identical and can be duplicated with a little more effort (and poorer performance) in Lingo. These methods are:

- mIsAlpha
- mIsUpper
- mIsLower
- mIsDigit
- mIsPunctuation
- mIsWhiteSpace
- mIsPrintable
- mIsGraphic
- mIsControl
- mIsHex

These are all Boolean functions (that is, they return either TRUE or FALSE), and all of them operate on a single character. I won't go into detail about all of them, but a few require a little more explanation. The mIsWhiteSpace method will return TRUE if the character in question is a tab character, a space or a carriage return. The mIsPrintable method will return TRUE if the character corresponds to one that would show up on a printer. A handful of control characters, such as the bell character [numtochar(7)], will return FALSE (since you can't print the beep sound), but in general this function will return TRUE most of the time. The mIsGraphic method indicates that the character is not one of the standard ANSI set of characters but is instead a graphical character used for drawing boxes or other shapes on a printer. The mIsControl method will return TRUE if the character's ASCII code is in the range 1 to 26; these characters are usually reserved by the system or the printer to do specific tasks such as create tabs (character 7), delete (character 8), do a printer line feed (character 10), carriage return (character 13) and so on.

The mIsHex method is most useful for ascertaining whether a given character falls within the range 0 to 9, A to F, which are the digits found in hexadecimal notation. For example, a simple hex-to-decimal conversion routine can be written as follows:

```
on HexToDecimal hexstr
   openxlib "movutils"
   set utils=movutils(mNew)
   set decnum=0
   set hexbase=1
   repeat with index=1 to length(hexstr)
      set hexchar=char (length(hexstr)-index+1)
of hexstr
      if utils(mIsHex,hexchar) then
         if utils(mIsDigit,hexchar) then
            set decnum=decnum+hexbase*hexchar
         else
  [design: note bold type in next line (all one
line)]
            set
decnum=decnum+hexbase*(chartonum(utils(mToUpperCase
,hexchar))-chartonum("A")+10)
         end if
      else
         set decnum=0
      exit repeat
      end if
      set hexbase=hexbase*16
   end repeat
   utils(mDispose)
   return integer(decnum)
end HexToDecimal
```

To use this routine, just pass HexToDecimal a hexadecimal number as a string, and it will return the decimal equivalent if the number is valid, and 0 if it is not. The highlighted line in the above code illustrates the engine that converts the nondecimal digits to hexadecimal—notice that it uses the mToUpperCase method to ensure that whether someone enters either HexToDecimal ("3ad4") or HexToDecimal("3AD4"), the results will be the same.

Boolean Operations

It occurred to me as I was writing the previous section on converting to hexadecimal that I was assuming that you know what hexadecimal notation is. When I was in college, I wrote an assembler for the 6502 chip on one of the school's new Apple IIe machines (which should uncomfortably date me); since it was an assembler, its purpose was to take mnemonic codes (shorthand for specific microprocessor instructions) and convert them into their hexadecimal equivalents.

Computers think in powers of 2, and hexadecimal is simply a notation for expressing four consecutive bits, or a nibble, in a form that we can use. The first 10 digits—0 to 9—are identical to decimals, but each digit from 10 to 15 is represented by a letter: A = 10, B = 11, . . . , F = 15. Thus, to represent the number 59:

1) Find the multiple of the largest power of 16 that is less than or equal to 59 (in this case 48, which is 3 * 16),

2) Subtract that number from the original,

3) Convert the multiplier to a hexadecimal number,

4) Repeat until the remainder is zero.

In our case, 59 - (3 * 16) = 11; since 11 corresponds to B in hexadecimal, the hexadecimal equivalent of 59 is 4B.

The assembler project took me about two weeks, most of it spent converting notation into hexadecimal bytes, entering them in a primitive text editor, and testing for bugs. I knew I needed to take a break when I woke up one evening in a cold sweat, with the dreadful realization that I had been dreaming in hexadecimal. I took up art for a while after that.

For the most part, being able to manipulate hexadecimal and binary numbers (numbers consisting solely of 0's and 1's) is no longer a prerequisite of successful programming; the development tools have become sufficiently powerful that most of the areas that do need such manipulation are handled automatically by the environments. However, there are still times when being able to manipulate binary numbers can come in handy.

One example from a project I worked on shows how bitwise Boolean operations can become an integral part of authoring. A children's product required a maze for a mouse to run through. The walls of the maze could shift, and I needed to find a mechanism that would allow for fast decryption of the information on user's machines, which could be slow.

I broke the maze into cells, where each cell was bounded by as many as four walls or as few as none. Arbitrarily choosing north(up) as the base line, I assigned each boundary a value: 1 for the north boundary, 2 for the east, 4 for the south and 8 for the west. Using this notation, it was possible to describe the state of each cell as a hexadecimal digit, with a 1 in each position indicating a wall and a 0 indicating a space. A cell that had walls on the north, south and west sides but none on the east would be represented by the number:

```
1 * 1 (north) + 0 * 2 (east) + 1 * 4 (south) +
1 * 8 (west)
= 1 + 4 + 8 = 13
13 in decimal = C in hexadecimal
```

The only direction that could thus be traveled to would be the one where the position = 0.

Until I discovered the Movie Utilities XObject, I ran into the following problem: While this notation was compact in the extreme and very fast, I was stymied if I wanted to do anything to manipulate the cells, since there were no bitwise Boolean operators. In normal Lingo, when you perform an "and" between two arguments, the processor will evaluate each expression and then say that if the expression equals 0, the expression is false, while if the expression does not equal 0, it's true. This is called a logical Boolean operation, in that the expressions are converted to logical values and then compared. In bitwise Boolean operations, on the other hand, each bit in one argument (assuming that the argument is an integer) is compared with the corresponding bit in the other argument, and the results are then placed in a third variable at that same position. For example, in the bitwise Boolean expression `1010 xor 1100`, each bit in the first number is compared with the bit in same position in the second number, and replaced by a 1 if the two bits have different values (e.g., 1 vs. 0) and by a 0 if the two bits are the same (say, both are 1). So `1010 xor 1100` gets broken down into four comparisons—(`1 xor 1`), (`0 xor 1`), (`1 xor 0`), (`0 xor 0`)—which equals 0110.

The Movie Utilities XObject offers four bitwise operators: mBitAnd, mBitOr, mBitXor and mBitNot. These work on long integers (i.e., integers 4 bytes long, the standard in Director), and return values as integers. So, to illustrate the above comparison: 1010 corresponds to integer 10, while 1100 is integer 12. Using xor on these two values together would work as follows:

```
put utils(mBitXor,10,12)
==> 6 - this corresponds to 0110 by some strange
coincidence.
```

The mBitNot method is the only one that doesn't take two arguments; it inverts the bits in a number, returning an integer in which values that had been 1 are now 0 and vice versa. In our cell example, if I wanted to find out whether it was possible to go east, I would invert the cell's bit values, then use BitAnd on them with the code for east: 0010, or 2 in decimal.

```
put utils(mBitAnd,utils(mBitNot,cell),2)
==> 2 - that is, since the code entered was the
same as the code returned, east is a viable direc-
tion
This can be broken down in binary notation as:

(not(1101) and 0010) ==> (0010 and 0010) ==>0010
```

A full working model of the maze is available on Hyperstand at http://www.hyperstand.com, and at http://www.blarg.com/~cagles/lingo.

There are several additional binary functions. The mBitSet function takes two arguments: a base number, and an index specifying which bit in that base integer should be set to 1. For example, utils(mBitSet,6,0) sets the "zeroeth" bit (the lowest bit) to 1; since 6 is binary 0110, utils(mBitSet,6,0) becomes 0111, or 7 in decimal. The mBitClear function sets the corresponding bit to 0. The mBitTest function will check to see if the indexed value is set to TRUE; it returns 1 (TRUE) if it is, or 0 (FALSE) if it isn't. Thus, utils(mBitTest,6,1) checks to see if bit 1 in the binary number is true (that is, 01**1**0); in this case it is, so the function returns TRUE. On the other hand, utils(mBitTest,6,3) (**0**110) returns FALSE, since the bit isn't set.

III mBitSet —Set specified bit within a long integer
III mBitTest —Test specified bit within a long integer
III mBitClear —Clear specified bit within a long integer
II mBitShiftL —Shift specified bit left within a long integer
II mBitShiftR —Shift specified bit right within a long integer
III mBitAnd —Perform logical AND operation of two long integers
III mBitOr —Perform logical OR operation of two long integers
III mBitXOr —Perform logical XOR operation of two long integers
II mBitNot —Perform logical NOT operation on a long integer
IS mBitStringToNumber —Translate string of '1's and '0's to a long integer

One of the advantages of using a base of 2 is that multiplying and dividing by 2 (both operations occur with astonishing frequency in computer programming

circles) become trivial. This is not true in general, since division especially can be computationally time-intensive. However, using bitwise operations, to multiply by 2 you only have to shift the bits of a number to the left by 1. Dividing (integer division here) by 2 is a corresponding move to the right. These shifts are facilitated in the Movie Utilities XObject by the mBitShiftL and mBitShiftR routines, which shift a long integer to the left or right by 1, filling in the open space with a 0. So, as an example:

```
   put utils(mBitShiftL,21) ==> mbitShiftL(10101)
==> (101010) ==> 42 decimal
   put utils(mBitShiftR,21) ==> mbitShiftR(10101)
==>(1010) ==> 10 decimal
```

Note that since you are dealing with integer division and multiplication here, shifting a number right and then left will only give you back the same number half the time (here, when the number is even). This routine can be useful in a wide variety of situations: binary searches, drawing routines and sorts are just a few.

Finally, there is a routine that can be used to convert a string of 1's and 0's to their decimal equivalents. While not especially useful, it can prove handy as a demonstration tool. This routine, mBitStringToNumber, takes as an argument a string and returns a long integer. It can also be handy as a debugging mechanism when dealing with any of the other bitwise operators.

Putting Away the Utility Belt

I've covered all of the functions of the Movie Utilities XObject but one: the mGetMessage routine. This is perhaps one of the most useful of all the functions in the belt, but its uses are so varied that there is no reasonable way to deal with it effectively in this article. So look for it soon in the *MUJ*, along with other ways of interacting with the system.

Because of the nature of these routines, there's a strong temptation to believe that some (though probably not all) of them will make their way into the next generation of Director Lingo in some form or another. These functions should all be reviewed if you come across a limitation in Director, before you reach into the XObject developer's kit to write a routine yourself. If you are like other developers, your time is usually at a premium, and reinventing the development language should not be a prerequisite to starting a new project.

Kurt Cagle (cagleK@aol.com) is a multimedia developer, writer and illustrator at the Center for Multimedia in Bellevue, Washington. He is the forum manager for MUJ *on Microsoft Network and is writing a book on Lingo programming for Random House.*

Authorware

Authorware Resources: Where to Go to Be in the Know

By Tom King

When I'm talking about Authorware at cocktail parties, people often flee in the other direction, but a few stragglers remain behind. [Editor's note: We generally do not recommend inviting Tom to parties unless you're hosting a gathering of insomniac instructional designers.]

After I've recounted the tribulations of a particularly tricky project in excruciating detail, one of the noncomatose observers usually asks, "Oh Wise One, how ever did you obtain this collection of semi-useful information?" As the other victims of my storytelling drift away, I disclose that, in fact, I discovered very little of this knowledge directly: I benefit from the experience of other Authorware developers by visiting Web sites, trolling for tech notes at electronic watering holes and consulting a few good old-fashioned books.

Rather than make you sit through one of my multimedia war stories, I'd like to share some of my resources.

Books: Ye Old Papyrus

Yes, boys and girls, there are paper-based books about multimedia, and even books about Authorware. There are even a few about multimedia or instructional design (shudder) in general.

***Computer-Based Instruction, Methods and Development*, 2nd ed., Prentice Hall. $50.** Steve Alessi and Stan Trollip provide a great overview of computer-based and computer-assisted instruction. Their book is a textbook at the graduate and undergraduate levels in many university instructional-design programs. It goes over the major categories of CBT such as tutorials, drill and practice, simulation, computer-based testing, and gaming activities. For each category the book discuss elements of good design, implementation issues and pitfalls to avoid. The chapter on authoring systems discusses Authorware as an example of an icon-based authoring system.

***Multimedia: Making It Work*, 2nd ed., Osborne McGraw-Hill. $34.95 with CD-ROM.** Tay Vaughan's book is a great reference to have handy, whether you are making a CD-ROM or working on a small in-house project. It includes tables and charts you never would expect to need but suddenly find incredibly valuable. For instance, there

are color plates with all of the standard palettes for Director, Windows and Macintosh. It also has a table listing all of the default fonts included with Macintosh System 7 and Windows 3.1, which can be *very* handy when working on cross-platform projects. When I'm not using *Multimedia: Making It Work* as a reference tool, I'll occasionally pick it up just to read a few of the project war stories for insights on the real-life experiences of other multimedia developers. The book is also available with a companion CD-ROM that includes demonstration versions of virtually every Macromedia product and some sample code and applications as well.

My Authorware Bookshelf Runneth Over

For a long time there was a disappointing lack of titles about Authorware. For starters, here are a few books that cover the basics:

Authorware Primer: A Guide to Features of Authorware 3.0 by S. Willard Elieson, University of North Texas. $35.

Authorware with Style by Roderick Sims. KnowledgeCraft (Australia), (61) 47-21-7211. Aus$49.95.

Authorware 2.0: An Introduction to Interactive Multimedia by P. Newhouse and R. Oliver, Napier Publications (Australia), (61) 9-361-3154. Aus$20 (plus Aus$10 for Mac and Windows companion disks).

The Multimedia Workshop: Authorware 2.0 by Matthew Holtz, Wadsworth Publishing. $20.

In addition to the primer, self-study and reference books listed above, there is a comprehensive three-volume reference work on Authorware by Joe Ganci. I have a copy of an earlier edition and recommend it highly to anyone just getting started. It provides detailed descriptions of every Authorware system function and variable. Each description includes examples, cross-references and suggested uses. The book has been expanded and updated to the extent that it has become separate volumes that may be purchased independently. Comprehensive updates covering Authorware 3.0 are due soon.

Authorware System Variables, Authorware Internal Functions and *Authorware External Functions* by Joe Ganci. Successful Multimedia. $65 per volume ($180 for the set).

Newsletters

I am aware (or shall I say http://aware) of two quarterly newsletters on Authorware, one of which my company, Solis Design, publishes. Both newsletters focus on Authorware specifically and include tips for everyone from neophytes to experts. *AuthorWords* is a shareware subscription newsletter; Solis will send you a free trial issue by mail, and subscriptions are $10. Back issues of *AuthorWords* are available as Acrobat PDF files on Hyperstand, the online home of *NewMedia* magazine and *MUJ* (http://www.hyperstand.com).

Authorware Intelligence Report, Successful Multimedia, 12471 Dillingham Square, Suite 122, Woodbridge, VA 22192; fax: (703) 878-2555, phone: (703) 878-2555.

AuthorWords, Solis Design, 107 South B St., Suite 350, San Mateo, CA 94401; fax: (415) 696-8703; phone: (415) 896-8700.

AWARE: The Ultimate Internet Source

AWARE is an Internet mailing list that unites the global Authorware community. Readers and contributors range from the designers and developers of Authorware itself to Net and multimedia "newbies." Along with the mailing list, the always kind and cheerful moderator, Danny Engleman, also maintains an AWARE World Wide Web site filled with useful information, hypertext links and goodies to download. For starters, try subscribing to the mailing list. Anyone with Internet access can do it by following these easy steps:

1) Create an e-mail message containing the following text on the first line: SUBSCRIBE AWARE <your name>.

Be sure to replace "<your name>" with the name you want other participants to know you as.

2) Send the e-mail to LISTSERV@CC1.KULEUVEN.AC.BE.

Shortly thereafter you will receive confirmation of your subscription and more instructions on how mailing lists work.

The next steps are optional, but *highly* recommended. They will cause a full day's worth of mail to be sent as one e-mail, rather than having individual postings arrive in your in box. (There can easily be 20 or 30 postings per day.)

1) Send a second e-mail to: LISTSERV@CC1.KULEUVEN.AC.BE

2) In the body of the message on the first line send the following text: SET AWARE DIGEST.

Again, you will receive a confirmation message, and future daily postings will be sent to you in a single file.

Acapulco Cliff Diving and the Web

While surfing the Web, I often feel like one of those Acapulco cliff divers familiar to anyone who has watched *Wide World of Sports.* From a vantage point high on a cliff (or your Internet service provider), you view a wide expanse of the sea of information. As you prepare to plunge into it, you feel exhilaration. However, as you begin diving, you feel nothingness; as empty air rushes past you, you might even feel fear—maybe you have chosen the wrong place to dive and there is nothing there. Here are a few Web sites that should have enough depth to keep you safe and happy:

AWARE (http://www.hvu.nl/aware): AWARE mailing list information, XCMD/DLL goodies and more.

Hyperstand (http://www.hyperstand.com): Home of *Macromedia User Journal* and *NewMedia* magazine, and electronic publications such as *Cyberlaw Journal* and *The Green Sheet*, an insider's look at new media. Look for the download section for multimedia toys.

Macromedia (http://www.macromedia.com): News, tech notes and other goodies.

Multimedia Information (http://viswiz.gmd.de/MultimediaInfo/): Index to multimedia information, from FTP sites to standards.

Solis (http://www.attila.com/solis.com): *AuthorWords* information, and product descriptions.

Successful Multimedia (http://access.digex.com/~mrmedia): *Authorware Intelligence Report* and more.

After the user-friendly Web browsing, you may want to venture into some other Internet multimedia and Authorware repositories. Here are a few FTP sites to explore:

AWARE (ftp://pa.cc.kuleuven.ac.be/pub/authorware): Authorware examples, DLLs, XCMDs, AWARE digests.

ITHACA (ftp://ithaca.icbl.hw.ac.uk/pub/authorware): Authorware example files.

Ohio University at Miami (ftp://m-media.mu.ohio.edu/M-Media): General multimedia tools and examples, including A3.

UNICORN (ftp://ftp.ccc.nottingham.ac.uk/pub/dlls_): DLLs and UCDs, including Authorware examples.

Macromedia Support

Of course, Authorware has had electronic support since long before the Web existed. Macromedia support is available on CompuServe (enter Go Macromedia) and America Online (keyword Macromedia). Of these two, CompuServe seems to have far more activity and attracts participants from all over the world.

Last but not least, Macromedia's tech support (415/252-9080) will handle questions about all Macromedia products; there are no longer separate numbers for different products. The main Macromedia number is (415) 252-2000.

I hope you find all of this information useful. I have tried contacting all the Web and FTP sites to verify the addresses, but some may change by the time you read this. Please send e-mail if you have comments, corrections or suggestions.

Tom King works at Solis Design Inc. in San Mateo, California. He can be reached at SolisInc@aol.com.

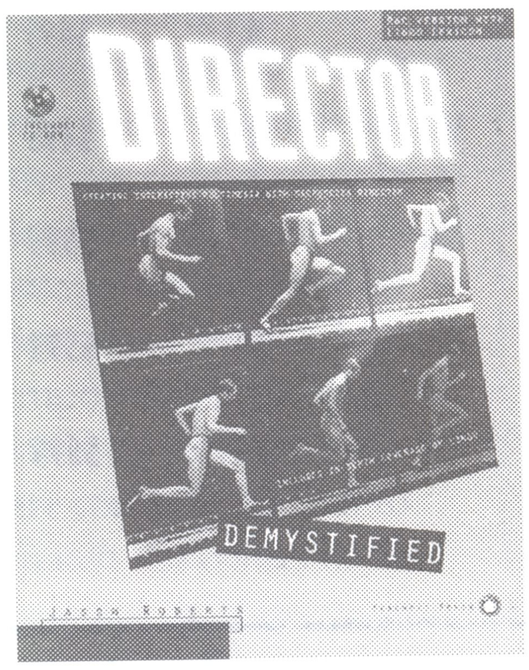
Resources

Director Demystified

By Eric Elia

Still looking for great Director resources to add to your bookshelf? *Director Demystified*, by Jason Roberts of San Francisco's Panmedia, is the first Director book we've seen yet to be considered truly thorough. Until now, we've seen great introductory books, like the *Visual QuickStart Guide: Director 4 for Macintosh* (*MUJ*, March 1995); great animation books like the *Macromedia Animation Studio* (*MUJ*, September 1995); and the first excellent Lingo book, *Macromedia Director Lingo Workshop* (*MUJ*, September 1995).

Weighing in at a hefty 776 pages and a relatively low price, *Director Demystified* ($39.95 from Peachpit Press, publishers of the *Visual QuickStart Guide*) had better be comprehensive—and it is. The book includes clearly written and well-organized chapters on sprite-based animation, Director's built-in auto-animation, interactivity and the basic Lingo need for such functionality, organizing your project before you crack Director and staying organized within the program, and using digital video and audio. Advanced chapters on the Paint window, ink effects, object-oriented programming—including handlers and Sprite, Frame, Cast and Movie scripts, and XObjects—are easily understood, even for new Macromedia users. The glossary and "Lingo Lexicon" are helpful reference chapters.



Also included is the requisite CD-ROM with tutorial movies, other unprotected movies for you to dissect, lots of clip media (including buttons, sliders and other controllers), shareware, demo software, an electronic version of the Lingo Lexicon and some XObjects too.

Peachpit Press (510) 548-4393; http://www.peachpit.com

Version Watch

November 1995

Product	Platform	Version	Price
Application Software			
Authorware (Includes Director Studio bundle)	Mac	3	$4,995
Authorware (Includes Director Studio bundle)	Win	3	$4,995
Action!	Mac	1.0.4	$199
Action!	Win	3.0	$199
Action! CD-ROM Edition (Action! SoundEdit Pro, ClipMedia 3)	Mac	—	$299
Action! CD-ROM Edition (Action! Turtle Beach's Wave Tools, ClipMedia 2)	Win	—	$299
Director ($29 upgrade for registered 4.0 Mac users)	PowerMac	4.0.4	$1,195
	Win	4.0.4	$1,195
Fauve xRes	Mac	1.2	$799
Fontographer	Mac	4.1	$495
Fontographer	Win	3.5	$495
FreeHand	Mac	5.5	$595
FreeHand	Win	5.0	$595
MacroModel	Mac	1.5	$895
MacroModel (includes Pixar's MacRenderMan)	Mac	1.5	$995
MacroModel (includes Macromedia Three-D)	Mac	1.5	$995
MacroModel (includes Pixar's Windows RenderMan)	Win	1.5	$995
SoundEdit 16	Mac	1.0.1	$379 (upgrades from SoundEdit, SoundEdit Pro and MacRecorder Sound Systems are $119; SoundEdit 16 replaces SoundEdit Pro and other earlier versions of SoundEdit)
X-Objects Developer's Toolkit (Windows version is included with Director for Windows 4.0)	Mac	N/A	$50

Director Studio bundle, $1,995 for Macintosh or Windows, includes Director, MacroModel, SoundEdit 16 (Mac), SoundForge (Win), and Adobe Premiere.

Graphic Design Studio bundle, $995 for Macintosh or Windows, includes FreeHand, MacroModel, Fontographer, Pixar RenderMan and Fractal Design Painter.

Product	Platform	Version	Price
ClipMedia			
ClipMedia 1: Business & Technology	Mac	2.0	$395
ClipMedia 2: Industry at Work	Mac	2.0	$295
ClipMedia 3: Perfect Presentations	Mac	2.0	$195
ClipMedia 1: Business & Technology	Win	2.0	$395
ClipMedia 2: Industry at Work	Win	2.0	$295
ClipMedia 3: Perfect Presentations	Win	2.0	$195

Coming Soon in the MUJ

- Tony Bové is back, and explains how to use the IMA-compressed audio format to get high-quality sound on CD-ROM.

- Chuck Henderson looks at what you need to know about Xres, and how it may fit into your development suite. Chuck will also take a closer look at the Macromedia Open Architecture and help you evaluate whether developing Xtras might be right for you.

- Tom King examines Rapid ST, a hot new Authorware-compatible rapid-prototyping and simulation tool for authors.

- Kurt Cagle takes a look at the Macromedia Common Scripting language. How will it differ from the Lingo you already use, and what will you need to know to get the most out of it?